Design - User Device Mapping

# **Background**

After successful login by the Admin person, user can onboard the new users along with their bio-metric and the devices [Bio, GPS, Scanner] to the machine, which is tagged to a particular Registration center. Through admin portal all the users, machines and devices are configured. But mapping of those users and devices to a particular machine would happen through this screen.

The **target users** are

* Super Admin
* Registration Supervisor
* Registration officer

The key **requirements** are

Device Mapping:-

On successful login of the user, will show off “New Registration” screen. There exists the “Menu”, which lists the options/ features. On clicking the “Device Mapping” link, should land up in displaying the list of devices along with the respective categories. Based on the user selected category [GPS, BIO, and Scanner] the respective devices specific to the enrollment center should be displayed in the left side list.

User can select the required devices [with specific serial no] and associate or de-associate with the current machine.

There is no device connectivity validation required.

The key **non-functional requirements** are

* Security:
  + Should not store any sensitive information as plain text information.
  + The data which resides in the data-base should be in encrypted format.
* Network:
  + Should able to communicate to the configured REST URL with proper authentication.
  + The http read timeout parameter to be explicitly set, if client unable to connect to the REST service.
  + Connectivity should happen through SSL mode. The respective key to be loaded during the call.

# **Solution**

The key solution considerations are –

**Service**

* On clicking the “Device Mapping” list in menu tab on Registration screen, it should hit the controller [**DeviceMappingController**] to fetch the master device and already mapped device list from database.
* **DeviceMappingController**
  + Invoke the interface **MachineMappingService** and call the **getDeviceMappingList(String centerId, String machineId, String deviceId)** to get the mapping list from db.
  + display the same to the screen at the left side panel of the screen
* Create **MachineMappingServiceImpl** and create DTO for the same.
  + When this service is triggered for **getDeviceMappingList**() functionality - it gets the device [master list and mapped list] details.
  + It hits DAO and the respective repository to fetch **list<master device> and list<mapped device>.**
  + Compare the whole device list with the already mapped device list and remove the mapped device from the master device list and render the result to the controller.
  + updateMappedDevice( List< DeviceMappingDTO>) – it does the device update to the db by calling the DAO class.
* **MachineMappingDAOImpl**
  + getAllDeviceBasedOnCenterId(String centerId, String deviceId) – return all the active device. If deviceid provided then filter the data based on the deviceId and render the result.
  + getAllMappedDevice(String machineId, String deviceId) - return all the device specific to the machine which are in active state. If deviceid provided then filter the data based on the deviceId and render the result.
  + updateMappedDevice( List< DeviceMappingDTO>) – this method updates the existing list in the database with the new list provided by the user.
* Send an Alert message (say) “User mapped successfully” or an error message.
* Handle exceptions in using custom Exception handler and send correct response to client.

**UI:**

* Design UI using FXML and map the UI individual components in DeviceMappingController class.
* DeviceMappingController – it should communicate between UI screen and Service ‘MachineMappingServiceImpl’ class to render the data to screen and capture the data from screen.
* Based on validation across the POJO class from the MachineMappingServiceImpl, build the UI screen
* Create the proper alert success/error to intimate the user.

**Classes**:

**UI**: DeviceMachineMapping.fxml

**Controller**: DeviceMappingController.java

**Service**: MachineMappingServiceImpl.java

**DAO**: MachineMappingDAOImpl.java

**Repository:** DeviceMappingRepository.java

**DTO**: DeviceMappingDTO.java

Class Diagram:

<TBD>

Sequence Diagram:

<TBD>